ELEC2870: Exercise Session 2
Multi-Layer Perceptrons

October 25, 2011

1 Objectives

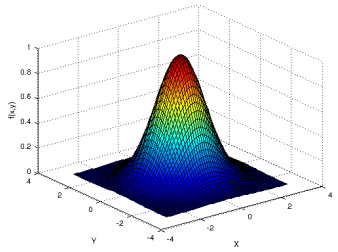
In this second exercise session, you will implement multi-layer perceptrons, i.e. neural networks with hidden layers. These models are much more powerful than single-layer networks, as they can learn complex non-linear relationships.

The objectives of this session are to (i) get familiar with multi-layer neural networks, (ii) implement the retropropagation algorithm and (iii) understand the importance of the number of neurons.

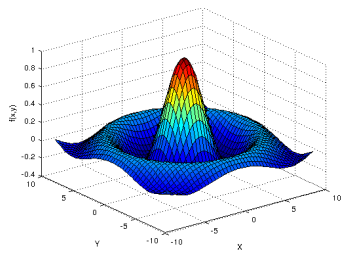
2 Learning Tasks

In this session, you will use MLPs to approximate two non-linear functions:

- a bell-shaped function $f : \Re^2 \rightarrow \Re : f(\mathbf{x}) = e^{-\frac{1}{2}\|\mathbf{x}\|^2}$;



- a wave-shaped function $f : \Re^2 \rightarrow \Re : f(\mathbf{x}) = \frac{sin\|\mathbf{x}\|}{\|\mathbf{x}\|}$.



In pratice, the analytical form of the function to approximate is unknown. The function should be approximated from a set of observations $\{(\mathbf{x}_p, t_p)|p = 1...P\}$, which may be polluted by noise and is called the learning set.

Usually, the performances of a model are estimated using an independent set of observations, which is called the test set. Indeed, it is important to check that the model behaves correctly, as you shall see in this exercice session.

Training and test sets are provided in an archive which can be found here: `http://moodle.uclouvain.be/course/view.php?id=84`.

Task : Visualise the training and test sets for both learning tasks. Compare them in terms of number of training samples, function smoothess, etc.

Tip : In order to visualise datasets, you can e.g. use the MATLAB function *plot3* and the function *visualise* which is provided in the archive.

3 Multi-Layer Perceptrons

Multi-layer perceptrons (MLPs) are multi-layer neural networks. Here, you will only use MLPs with one hidden layer, which is enough for continuous functions.

Question : MLPs are universal approximators, what does it mean ?

Multi-layer perceptrons generalise single-layer neural network. They consists of several successive layers of neurons, where the output of a neuron can be transmitted to the input of one or more neurons of the next layer. Figure 1 shows an example of MLP with one hidden layer, which are used in this session.

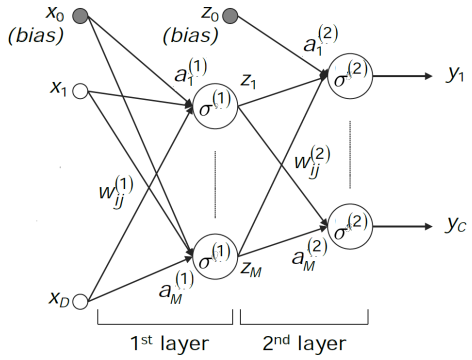


Figure 1: Multi-layer perceptron with one hidden layer.

Let us quickly review the notations for MLPs with one layer of M hidden neurons and only one output neuron. For the lth layer, $\mathbf{W}^{(l)}$ is the matrix of weights, where $\mathbf{W}_{ij}$ is the weight from neuron i to neuron j. The weigth matrix $\mathbf{W}^{(1)}$ is a $D \times M$ matrix, whereas the weigth matrix $\mathbf{W}^{(2)}$ is a $M \times 1$ matrix.

If $\mathbf{x}$ is an input row vector of dimension D, the output of the hidden layer is the row vector $\mathbf{z} = \sigma^{(1)}\left(\mathbf{x}\mathbf{W}^{(1)}\right) = \sigma^{(1)}\left(\mathbf{a}^{(1)}\right)$ of dimension M, where $\mathbf{a^{(1)}}$ is the activation of the hidden layer. The output of the network is the value $y = \sigma^{(2)}\left(\mathbf{z}\mathbf{W}^{(2)}\right) = \sigma^{(2)}\left(a^{(2)}\right)$, where $a^{(2)}$ is the activation of the output neuron.

Similarily to Session 1, the bias is systematically added to the neuron input. In this session, the activation function for hidden neurons is

$$\sigma^{(1)}(a) = \frac{1}{1 + exp^{-a}}, \tag{1}$$

whereas the activation function for the output neuron is

$$\sigma^{(2)}(a) = a. \tag{2}$$

Question : Why is $\sigma^{(1)}(a) = \sigma^{(2)}(a) = a$ not a good idea ?

Task : Implement a perceptron with one hidden layer and one output. Make sure that the number of hidden neurons can be easily modified.

Tip : Start by writing code to compute (i) $\mathbf{z}$ from $\mathbf{x}$ and (ii) y from $\mathbf{z}$.

Task : Build a few MLPs with different number of neurons and random weigths. Then, visualise their outputs for the test samples (without learning).

Tip : In order to visualise MLP predictions, you can e.g. use the functions *scatter*, *plot3* and *visualise* (which is provided in the archive).

4 Error back-propagation

The learning phase of MLPs consists in minimizing an error criterion by adaptation of the weights. For a regression problem, it is common to choose the least square criterion

$$E = \frac{1}{P} \sum_{p=1}^{P} (y_p - t_p)^2 = \frac{1}{P} \left[\mathbf{y} - \mathbf{t}\right]^T \left[\mathbf{y} - \mathbf{t}\right]. \tag{3}$$

Several methods based on the gradient descent can be used to minimize this criterion. Here, you shall focus on the error back-propagation method, which is described in the course. The error back-propagation algorithm allows computing the gradient of the error in a recursive way.

Task : Implement the error back-propagation algorithm to compute the gradient of the error. Store the gradient for each layer in a different matrix with same shape than the corresponding weight matrix.

Once the gradient is computed, you can use a simple gradient descent to update the weights of a MLP. The formula for weights adaptation is still the same

$$\mathbf{W}_{k+1}^{(l)} \leftarrow \mathbf{W}_{k}^{(l)} - \alpha_k \nabla_{\mathbf{W}^{(l)}} E_k. \tag{4}$$

Task : Implement a gradient descent to learn the weights of a MLP using a given number of hidden neurons and a training set.

Task : Test your implementation on each learning task using a few neurons:

- learn a MLP from training samples,
- check that the error criterion decreases during gradient descent,
- visualise the output of your network for test samples and
- compute the error made by the MLP on test samples.

5 Network Architecture

The number of neurons is an important meta-parameter of multi-layer perceptrons. Depending on the number of neurons, an MLP may be able to solve a learning task or not.

Task : For each learning task, compare the results obtained by MLPs with different number of neurons, e.g. 1, 2, 3, 4, 5, 10, 20 and 100 neurons.

Tip : Models can be compared based on visualisation and test errors.

Question : Which problem arises when a MLP has not enough neurons ?

Question : Which problem arises when a MLP has too many neurons ?

Question : Is the optimal number of neurons identical for both tasks ? Why ?

Question : Which criterion could be used to select the number of neurons ?